# Closed Set Text Independent Speaker Identification using Kernel Machines

*By:* Bhusan Chettri

*Project Supervisor:* Prof. Thomas Hain

*Programme:* MSc Computer Science with Speech and Language Processing

This report is submitted in partial fulfillment of the requirement for the degree of MSc in Computer Science with Speech and Language Processing.

September 10, 2014

## Declaration

All sentences or passages quoted in this dissertation from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations which are not the work of the author of this dissertation have been used with the explicit permission of the originator (where possible) and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this dissertation and the degree examination as a whole.

Name: Bhusan Chettri

Signature:

Date: September 10, 2014

**Abstract**

Speaker Identification is the task of identifying whether a particular person exists in the enrolled set of speakers or not by using their voice characteristics. One of the key challenges however is to efficiently deal with the channel and environment variability in the speech signal to improve the identification performance. Here we deal with closed set text independent system which means that the users must be an enrolled user of the system and there is no constraint in the text being spoken during enrollment and testing.

The main aim of the project is to implement Speaker Identification System based on Kernel Machines and extend the concept by Frame Concatenation to include more wider temporal context and hence enhance the identification accuracy. The system is trained on 50 target speakers and performance is evaluated on 150 test utterances. The baseline GMM-UBM system is built using ALIZE toolkit to compare the performance of our system. Three different variants under this Frame Concatenation approach have been explored in the project. However none of our System could outperform the baseline GMM-UBM accuracy of 57.3%.

**Acknowledgements**

I would like to express my sincere gratitude to my supervisor Dr. Thomas Hain, Professor, Department of Computer Science, University of Sheffield for giving me an opportunity to do my Master's Dissertation under his guidance. His consistent guidance and feedback throughout the project has been remarkable which helped me progress and complete my dissertation on time.

Also, I take this opportunity to thank Mauro Nicolao, Research Associate, SPandH Research group, Department of Computer Science for his time and patience in listening my queries and giving useful suggestion. His help for building baseline system has saved a lot of my time which i could spare doing the research work on my project.

I am also very grateful to Erfan and all other PhD students of Prof. Thomas Hain working in the MINI lab for being so kind and helpful to me at all the times during my dissertation.

Last but not the least, i would like to thank my parents for their consistent support and prayer for my studies and good health that helped me complete this Master's Program in Computer Science at the University of Sheffield.

# List of Figures

# List of Tables

# Contents

# Chapter 1

# Introduction

Speech is the primary means of communication among humans that delivers many level of information. It conveys linguistic information like message being spoken, language and speaker information (eg gender, emotion, physiological characteristics). The speech signal also conveys information about the environment where the speech was recorded/produced and transmitted. Speech Recognition algorithms are more concerned with creating models that can recognize the message present in the speech signal with high accuracy while Speaker Recognition algorithms are dedicated towards developing robust algorithms to model the speaker more accurately there by achieving higher recognition rates. Though many algorithms and techniques are available, Polynomial Kernels have been used in this project for Speaker Identification. This chapter discusses some of the basic concepts in Speaker Recognition.

## 1.1 Speaker Recognition

Speaker recognition is the task of identifying or verifying the person by using the characteristics of their voices. The task can be broadly divided into two major categories: Speaker Verification or authentication and Speaker Identification [48]. Speaker Verification is the process of verifying whether a person is actually that person who he/she claims to be or not, on the basis of a sound recording, typically of length 5-60 seconds. Naturally prior recordings of speakers to be recognized have to be available, but often these are short too. From these so-called speaker models are derived and stored. Recognition involves comparing the recording of the claimed person with the particular claimed speaker model in the system. In contrast, Speaker Identification [48] is the process of identifying whether the particular person exists in the enrolled set of speakers. It involves comparing the claimed speaker against the pool of enrolled speaker models in the system.

Speaker Recognition systems are further divided into two types [34], text dependent and text independent systems. In text dependent systems, the text being used during the enrollment and recognition must be same. This means that during the enrollment phase users are asked to input their voice with respect to some predefined text into the system

for which speaker models are created. During recognition phase they are prompted to read the same text that was used during the enrollment [34][48]. However, with text independent systems, their is no restriction with the words being spoken during enrollment and recognition phase. The set of words spoken during enrollment and recognition can be completely different. Thus this task is quite challenging and difficult in comparison to text dependent systems.

Since we are dealing with closed set identification, the user must be an enrolled user of the system. However in case of an open set identification problem the system must be capable of determining whether the user belongs to one of the enrolled user or not. This means that, it involves first determining whether the user is one of the registered user from the pool of users. The second stage is only carried out if the user is one of the enrolled user else he/she must be rejected as an impostor.

The performance of a Speaker Recognition system can be measured differently for different task. As the output of a Closed Set Speaker Identification system is a speaker identity predicted from a set of enrolled speakers in the system, the identification accuracy [46] is used to measure the performance which is defined as

$$identification\ accuracy = \frac{\#\ correctly\ identified\ segments}{total\ \#\ of\ segments\ tested} * 100 \tag{1.1}$$

On the contrary with Speaker Verification systems performance is measured in a different manner. There are two types of error associated with verification systems: false acceptance and false rejection [20]. False acceptance error occurs when an impostor is accepted as a speaker and false rejection is a situation when a true speaker is rejected by the system as an impostor. Therefore the overall performance of a such systems is defined in terms of Equal Error Rate (EER) [20][42] which is a situation where false rejection equals false acceptance as shown in following equation.

$$EER = False\ Acceptance = False\ Rejection \tag{1.2}$$

## 1.2 Kernel Machines

A Kernel Machine is basically characterized by a Kernel Function k(.,.) [4] that represents a measure of similarity between two vectors through a simple inner product in a high dimensional feature space (also called Kernel Space). The following equation shows a Kernel Function [4]

$$\mathtt{k}(x, y) = \langle \phi(x), \phi(y) \rangle \tag{1.3}$$

where $\mathtt{k}$ is the Kernel, $x$,$y$ are the input vectors and $\phi$ is a mapping function that transforms the input vector $x$ and $y$ from input space to a high dimensional Kernel Space. These Kernel Function have been widely used with Support Vector Machines for performing binary classification [17]. With such Kernel Function the data set which is non linear in

the input space becomes linearly separable in the higher dimensional space. This is one of the key and fundamental property that makes Kernel Machines very powerful.

The dimension of the feature vector in the Kernel Space is very large, even infinite as in the case of Radial Basis Kernel (RBF)[13][49]. So practically and computationally this becomes challenging and normally not used. However the solution to this problem is by applying the so called **'Kernel Trick'** [13]. With this trick we can get all the task done that were suppose to be carried out in this high dimensional Kernel Space without actually visiting this Kernel Space. In other words we would not require to do the actual transformation to kernel space. The Kernel Function shown below

$$\mathtt{k}(x, y) = \left(x^T y + c\right)^d \tag{1.4}$$

is a Polynomial Kernel of degree d [4]. The Kernel Trick in this equation allows Kernel Function to measure the similarity between the two vectors with a simple dot product and then by raising the result to the power of d, we get our required result. All this computation is carried out in input space without ever having to perform Polynomial Expansion into higher dimensional space.

## 1.3 Objective

The aim of the project is to investigate direct Kernel based Speaker Identification Systems and to explore the extension of existing regimes to neighboring context relations. The system being developed is a Closed Set Text Independent System where recognition operation can be performed only on enrolled set of users and there is no restriction in the words being spoken during enrollment and recognition. In the first phase of the project the regime to derive Polynomial Kernels as described in [14] should be implemented using Python. The performance should be tested on a subset of the NIST SRE 2008 evaluation set that allows reasonable assessment while constraining computational complexity. Adequate GMM-UBM baselines need to be obtained. In a second stage the Kernels should be extended to wider temporal contexts. This can be achieved for example by extension of Kernels to neighboring frames, i.e concatenation of neighboring frames. An alternative is indirect context extension through deep neural network front-end feature computation. The computational complexity at this point is potentially large, and will require grid computing.

## 1.4 Overview of the Report

In this chapter we have briefly described the basic concepts in Speaker Recognition. Chapter 2 shall cover a detailed literature on Speaker Recognition. In Chapter 3 we will describe the main methodology used in this project that is based on Kernel Machines for Speaker Identification. Chapter 4 briefly discusses various steps that were used to build

the GMM-UBM baseline system through the ALIZE toolkit. Chapter 5 describes different experiments that were performed throughout the project along with the experimental results. Chapter 6 finally concludes the report.

# Chapter 2

# Literature Review on Speaker Recognition

In this chapter we shall discuss the literature on Speaker Recognition starting with the features that are commonly used for modeling target speakers, different modeling approaches that exist and has been explored so far in the field. We shall also briefly discuss the history/evolution of the technology, various problems and challenges in Speaker Recognition and different open source tools and softwares available for research in this field.

## 2.1 Introduction

The idea behind making computers recognize human by their voice, so called Speaker Recognition was initiated and encouraged in the early 1960s with the work of authors in [22] at bell laboratories where they built an isolated digit recognition system using the formant frequency that was measured over the vowel regions of each digit. This particular work motivated and encouraged many researcher to take up research and development in this technology and progress towards making the system more robust and enhance the recognition performance. Eventually over the period of last ten years the technology has really progressed as evident from various Speaker Recognition Systems presented to the National Institute of Standards and Technology (NIST) [25] that provides a common platform for computing the performance of the Speaker Recognition Systems.

## 2.2 Features

The first and foremost step in any Automatic Speech Recognition system is called feature extraction phase which is commonly referred as the "Front End". This is often called as the pre-processing step where the raw speech signal is converted into a sequence of acoustic feature vectors that represents some specific information about the speech signal.

The most commonly used acoustic features for Speech Recognition are Mel Frequency Cepstral Coefficients (MFCC) [23], Linear Prediction Cepstral Coefficients (LPCC) [9][36], and Perceptual Linear Prediction Cepstral Coefficients (PLPC) [30][41]. These features represent spectral information that is obtained through short time windowing of speech segments.

MFCC features are directly derived from the FFT (Fast Fourier Transformation) power spectrum whereas the LPCC and PLPC uses an all-pole model [36][30] to represent the smoothed spectrum. The centers and bandwidths of mel-scale filterbank are based on the mel-frequency scale shown below

$$m = 2595 \log_{10}(1 + \frac{f}{700}) \tag{2.1}$$

where f = frequency in hertz and m = frequency in mel scale.

It gives more emphasis on lower frequencies and thus matches the idea behind human perception of the speech signal. The filterbank spectral representation are transformed to cepstral coefficients by taking discrete cosine transformation (DCT). Only DCT coefficient 2-13 are kept making a 12 dimensional MFCC vector [23] while discarding all other coefficients including the first coefficient representing the energy component.

Among the different types of features PLPC and MFCC are mostly used in Automatic Speech Recognition Systems [37]. Though authors in [9] have shown application of LPCC features in Speaker Recognition but currently MFCC have found more wider usage in Speaker Recognition applications as evident from the fact that the systems presented in 2008 SRE evaluation were mostly based on MFCC features [25].

## 2.3 Speaker Modeling Techniques

In this section we shall be briefly discussing some of the classical modeling techniques that has been used in the field of Speaker Recognition.

### 2.3.1 Vector Quantization (VQ)

These models are commonly referred as non parametric models where a set of feature vectors belonging to each speaker are compressed into a single cluster of phonetic sounds referred as a VQ codebook [27]. This codebook is then used to model each speaker. Thus a template for each speaker model is created. Recognition procedure simply proceeds by matching the stored speaker template against the unknown test utterance. However these models fail to capture the variabilities that can come in an unconstrained speech task.

### 2.3.2 Gaussian Mixture Model (GMM)

Literature [46][42] shows that gaussian mixture model (GMM) forms the core of the state of the art technique in text independent Speaker Recognition. GMM is a stochastic model

[42] based on an assumption that all the feature vectors follow a gaussian distribution that is characterized by mean and the variance. GMM can be thought of as an improvement over the VQ method where the clusters are overlapping. In other words GMM is based on soft clustering approach where each feature vector has a non zero probability of being generated from any one of the gaussian component rather than being assigned to the nearest cluster as in the case with VQ models.

A GMM is a weighted sum of K component gaussian densities as shown in the equation below

$$\mathtt{p}(\vec{x}|\lambda) = \sum_{i=1}^{K} w_i b_i(\vec{x}) \tag{2.2}$$

where $\vec{x}$ is a $D$-dimensional feature vector, $\mathtt{b}_i(\vec{x})$, $i = 1, ..., k$ are the component gaussian densities and $w_i$, $i = 1, ..k$ are the mixture weights [46].

Each component density is a D-variate gaussian function of the form

$$\mathtt{b}_i(\vec{x}) = \frac{1}{(2\pi)^{D/2}\left|\Sigma_i\right|^{1/2}} exp\left\{-\frac{1}{2}(\vec{x}-\vec{\mu_i})^{'}\Sigma_i^{-1}(\vec{x}-\vec{\mu_i})\right\} \tag{2.3}$$

with $\vec{\mu_i}$ and $\Sigma_i$ as the mean vector and covariance matrix respectively. Also the mixture weights satisfy the constraint $\sum_{i=1}^{k} w_i = 1$.

Therefore a speaker model using GMM is represented by three parameters: mean vectors, covariance matrices and mixture weights from all component densities. These parameters are often represented using the notation shown below [46].

$$\lambda = \{w_i, \mu_i, \Sigma_i\}, i = 1, 2, ....K \tag{2.4}$$

The above model parameters are estimated using Expectation Maximization (EM) algorithm [11] that ensures a monotonic increase in the likelihood value of the data. More details about the GMM approach can be found in [46].

The authors in [46] have addressed various performance issues related to the length of the training data and number of gaussian components that is essential to maintain a good recognition performance. They claims that minimum 16 gaussian components is required to effectively model speaker characteristics and minimum 1 minute training data is essential to get a good recognition performance, below which the performance degrades dramatically.

### 2.3.3 Universal Background Model (UBM) and GMM-UBM

The universal background model, UBM is commonly referred as world model which is an independent speaker model being trained with lots of training data (hours of speech) from large number of different speakers in order to capture the general characteristic of a speaker [45]. The optimal value of these model parameters are estimated using the EM algorithm [11]. During speaker enrollment as shown in Figure 2.1 the parameters of the

background model (UBM) are adapted using the speaker specific feature distribution. Hence the adapted GMM-UBM model is further used as the speaker model which is more representative and reliable than an individually trained GMM.

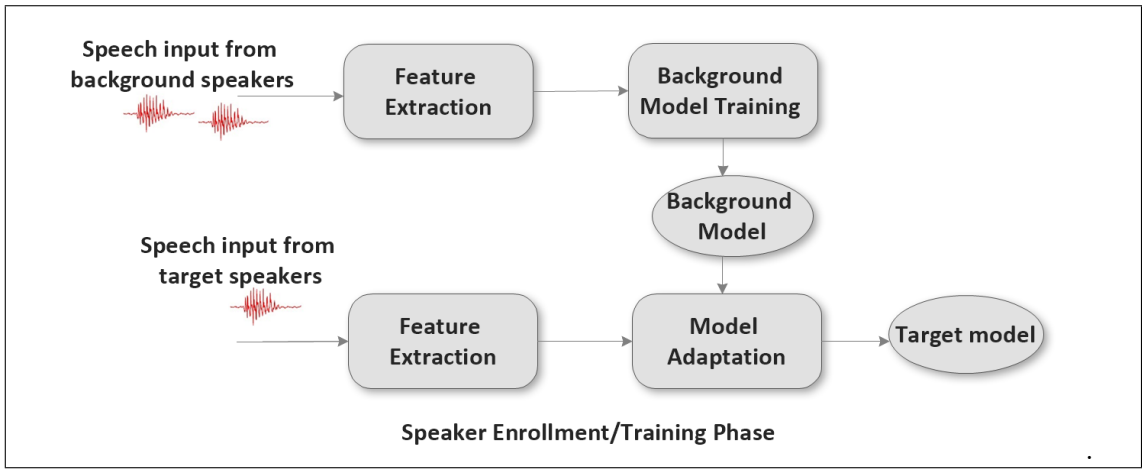


**Figure 2.1:** Enrolling Target Speakers into the System [34]

This methodology of adaptation solves the problem of data sparsity with the target speaker, because many often we have very less training data for the target speakers. This approach makes the model more powerful and perform better to unseen data as it acquires the general attributes from the underlying UBM model [48]. The most commonly used adaptation method to derive a speaker specific GMM from the background model is the maximum a posteriori (MAP) adaptation [11]. Literature shows that adapting means only from the speaker data works well while the other parameters are common to all speaker models [45].

During recognition as shown in Figure 2.2, the score for the unknown test utterance is computed using both the MAP adapted speaker model and the UBM (impostor) model. The final score is defined as the log likelihood ratio between the target score and the background score [45][34] as shown below

$$\Lambda(\mathtt{X}) = \mathtt{log\ p}(\mathtt{X}|\lambda_{SPK}) - \mathtt{log\ p}(\mathtt{X}|\lambda_{UBM}) \tag{2.5}$$

where $\mathtt{log\ p}(\mathtt{X}|\lambda_{SPK})$ is the log likelihood score of test utterance $\mathtt{X}$ using target model and $\mathtt{log\ p}(\mathtt{X}|\lambda_{UBM}$ is the score obtained through background model.

There is a slight difference in the way scoring is performed with respect to Identification and Verification. Verification [34] is basically a binary classification task where only the particular target model is used to compute the score for the claimed identity. If the score is above the defined threshold then the speaker is accepted as a valid user else it is rejected as an impostor [15].

On the contrary identification involves finding the best matching model for the unknown test utterance, hence the task is called a multi class classification problem. Here score is obtained using each of the enrolled models in the system and the model for which

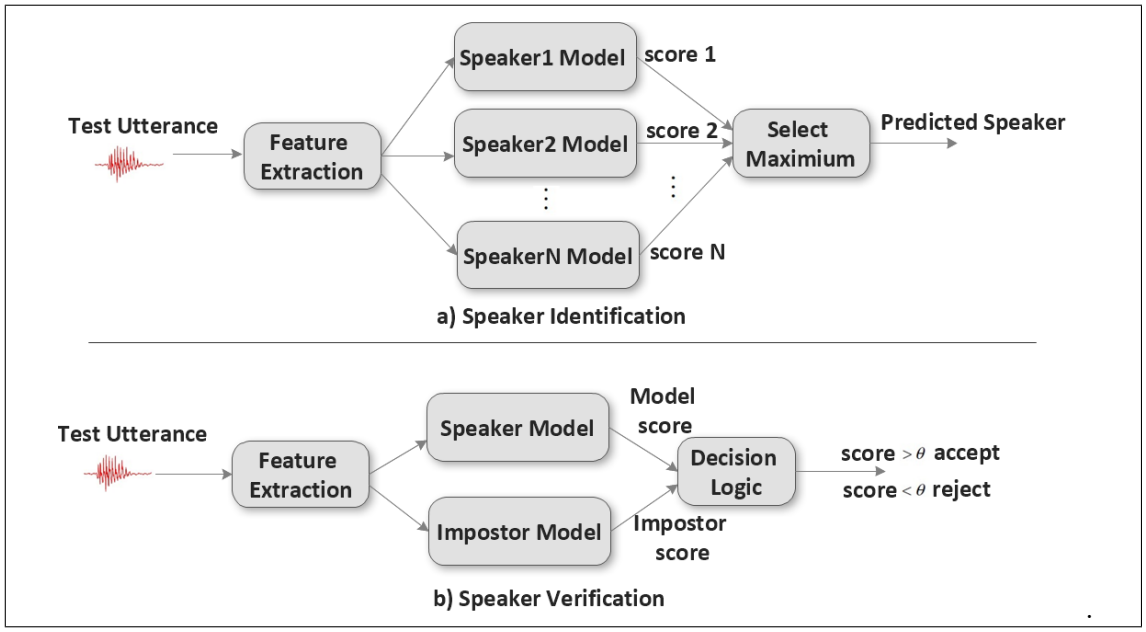


**Figure 2.2:** Components of a Speaker Recognition System [34]

a highest score is obtained [48][15], becomes the identified speaker of the unknown test utterance [45].

### 2.3.4 Support Vector Machines

Support Vector Machines (SVM) are discriminative classifiers [17] that has been widely used in Speaker Recognition because of its ability to strongly distinguish between two classes of data, (Speaker and the Impostor) with good precision and accuracy. SVM transforms the input space into a higher dimensional feature space and then separates the two classes with an optimal hyper plane [49][13]. Unlike training generative models (GMM) which requires only set of training examples for building speaker models or the background models (UBM), training the discriminative model needs training data from both the target speaker and the impostor [17].

The SVM shown in Figure 2.3 is basically a binary classifier that finds a seperating hyperplane between two classes as a decision boundary. The hyperplane is chosen in such a way that the distance between the nearest vectors and the hyperplane in both sides are maximized [18]. This criterion of finding an optimal hyperplane is often called maximum margin criterion and the closest vectors on each side are called the Support Vectors [18][17]. The classifier discriminant function [17] is written as

$$f(x) = \sum_{i=1}^{N} \alpha_i t_i K(x, x_i) + d \tag{2.6}$$

where $x_i$ are the Support Vectors, $t_i$ is the class output labels that could be either +1 or -1 (+1 for speaker class and -1 for the impostor), $\sum_{i=1}^{N} \alpha_i t_i = 0$, and $\alpha_i > 0$. These parameters along with the bias term $d$ are estimated from the training data set through

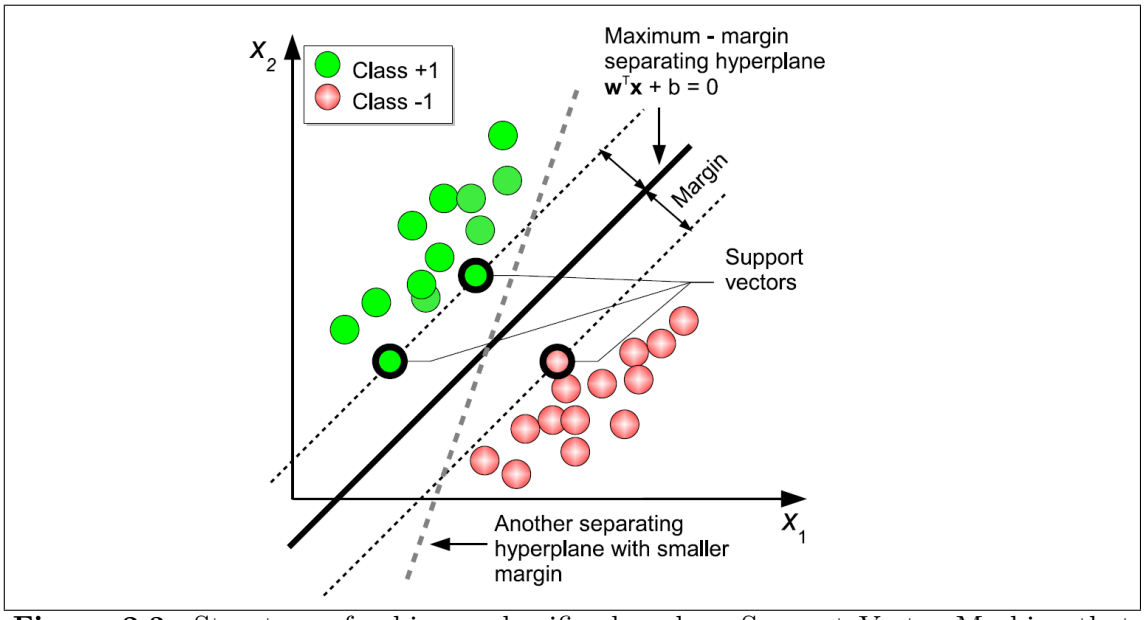


**Figure 2.3:** Structure of a binary classifier based on Support Vector Machine that seperates two classes of data (+1 and -1) with an optimal separating hyperplane with the help of support vectors [34].

some optimization process [17]. Here the Kernel Function $\mathtt{K}(.,.)$ is defined as a mapping of the input feature space into a high dimensional Kernel Space, expressed as

$$\mathtt{K}(x, y) = \mathbf{p}(x)^t \mathbf{p}(y) \tag{2.7}$$

where $\mathbf{p}(\mathrm{x})$ represents the transformed vector x in higher dimensional space (could be an infinite dimension). The non linear data in input space that was difficult to classify becomes linear in a higher dimensional space and hence classification becomes easier. This is the main reason why Support Vector Machines is considered to be a better classifier among other binary classifiers.

One of the key feature of this classifier is the ability to achieve a very high degree of performance with much lesser training data compared to GMM-UBM [18]. The generalization ability of SVM to unseen data makes it more popular and a powerful classifier [18]. When performing Speaker Recognition the speaker utterance will have variable sequence of feature vectors, however SVM requires these sequence of vectors to be represented by a single static data vector suitable for doing classification. Some of the approaches to do this transformation is by making use of Polynomial Classifiers [49][16], Sequence Kernel functions as described in [17] and GMM supervectors [18].

In case of Speaker Verification we basically train a target class with all vectors labelled as +1 and -1 for the impostor class. The optimal hyperplane is found that optimally separates this two classes. Score is then computed for each vector in the test utterance based on whether they are classified as target or the impostor. If the score is above a defined threshold then the speaker is accepted as a valid user else it is rejected as an impostor.

As SVM is a binary classifier [18], to deal with Speaker Identification problem which is actually a multiclass problem, one vs all training criterion is followed where each target model is trained against all the remaining N-1 speakers. N binary classifiers are trained in this way for N target speakers. During classification, a score for the test utterance is obtained from each of these N classifiers and the classifier for which maximum score is obtained is selected as the predicted speaker of the test utterance.

### 2.3.5 i-Vectors

The supervector for a speaker that consist of speaker dependent mean components is normally extracted by performing a MAP adaptation on the background model (UBM) [34]. However this supervector not only contains speaker specific information but other factors like channel variations and noise are also included during the adaptation process. Joint Factor Analysis (JFA) [32] provides a mechanism to deal with the inter speaker and channel variability thereby enhancing the performance of the Speaker Recognition System. The intuition behind JFA is that the speaker dependent supervector can be expressed as a combination of following four terms [24]

$$M = m + Vy + Ux + Dz \tag{2.8}$$

where $m$ is a speaker-independent supervector (from UBM), $V$ and $D$ are speaker dependent subspaces that represents the eigenvoice matrix and the residual matrix (diagonal) respectively and $U$ represents the channel subspace (eigenchannel matrix) [33]. The vectors $x$, $y$ and $z$ denotes the speaker, channel and speaker-specific residual factors respectively.

Speaker Recognition using JFA technique requires computation of these subspaces ( $V$, $U$ and $D$) and then further estimating the factors $x$, $y$ and $z$ for a given training utterance of the target speaker [32]. The channel dependent subspace and channel factor (term Ux) is then removed from the above equation (2.8) to obtain a channel/session compensated speaker model (i.e, $M$ - $Ux$). Score is then obtained by computing the likelihood of the unknown test utterance using this new model [24].

However one drawback with JFA is that the number of dimension after decomposition of the supervector is very high. The concept of i-Vector has been proposed [24] to deal with this dimensional issue and currently this i-Vector based system has been the state of the art technique in Speaker Recognition [24]. An i-Vector is a compact representation that combines the speaker and channel subspace into a single low dimensional subspace called the 'total variability space'.

Therefore the supervector $M$ using i-Vector approach is expressed by the following equation

$$M = m + Tw \tag{2.9}$$

where $m$ is same as in equation (2.8), $T$ is a low rank rectangular matrix called the total variability matrix and $w$ is the i-Vector which is of very low dimension that represents

speaker specific information [24].

### 2.3.6 Others

Many other different techniques like Hidden Markov Models (HMM), Neural Networks (NN) have also been used in the literature for Speaker Recognition. HMM is a stochastic model that has been very successful in Speech Recognition because of its ability to model the temporal information in the speech signal. HMM has also been used in Speaker Recognition [38] but it has not been as much successful as it is with Speech Recognition.

Neural Networks [47] on the other hand are discriminative models that is based on modeling a decision function which can be used to discriminate well between a target speaker and an impostor. NN has been used in different forms [47][39] for solving different types of task, however one key problem with the NN is the overhead of training the entire network all over again whenever a new speaker model is to be added to the system. On the contrary this is not the case with the Polynomial Classifiers as described in [16] where entire models need not be re-trained when a new speaker is to be added. The procedure just requires computation of the new speaker model and the existing pool of speaker models is updated with the new model [16].

## 2.4 Problems, Challenges and Solution

The main challenges and problems that is often encountered by the research community in Speaker Recognition is in making advancement and progress towards achieving more robustness and accuracy under various real time situations. Some of the factors that influence this are quality of the speech signal being used to train and test the systems, length of the training and test utterances, size of the target population being modeled (in case of identification this is an important factor) and also the phonetic content of the speech signal.

### 2.4.1 Quality

The quality of the speech signal being used to train and test the speaker models plays a very significant role in recognition performance [27]. Models trained with noisy data performs poorly during recognition phase. If the noise in the training data are not taken care of properly before training the target speakers then the resulting model will not only represent speaker specific properties/attributes but would also represent the background noise. This means that during classification these models would not be able to discriminate with 100% confidence between a true speaker and an impostor [34]. As a result we find degradation in the accuracy of the recognition system. Similarly if models were trained without much noise in it (sort of clean speech) and testing performed on test utterances containing some background noise then also models would not perform very well due to

the presence of noise and silence frames in the test utterances [34]. As a consequence performance in such case also would be very poor.

### 2.4.2 Length

Another important contributing factor that affects the performance of the Speaker Recognition system is the duration/length of speech signal used for training and testing. There must be sufficient training data to train the target models because with more training data the models will be able to learn more wider phonetic content and hence would be able to give good recognition performance [46] during testing. In other words as we train our models with more and more data, generalization ability of the models to unseen data would increase thereby increasing the recognition accuracy on unseen test utterances. Similarly length of test data is also very important to have a good recognition accuracy. With very small length test data there are high chances of misclassification because there is very less information supplied to the recognition system which would find it difficult to discriminate between the target speaker and the impostor with enough confidence. Reynolds et al.(1995) [46] have shown how different length training and test data affects the performance of the Speaker Identification System based on Gaussian Mixture Models.

### 2.4.3 Cross Channel

The performance of the Speaker Recognition System would be affected adversely when there is a channel mismatch between the training data and the test data as it introduces spectral differences and distortions in the speech signal [34]. This condition could be because of use of different microphones and different transmission channel (cellular phones, telephone speech). Therefore it is very important to ensure that proper channel compensation methods are used before training and testing on both the data sets to eliminate this channel variation effect. This would help improve the performance of the system else it can dramatically degrade the recognition accuracy [48].

### 2.4.4 Population Size

The number of target speaker being modeled also directly affects the performance of the recognition system especially with the Speaker Identification Systems [46]. With the increase in the population size the accuracy of Speaker Identification Systems degrades because the probability of misclassification increases as there is an increase in the number of comparison required to be made against each of the enrolled models in the system [48].

### 2.4.5 Solution

Following approaches/methods are commonly employed to deal with some of the problems that were discussed in the previous sections in order to make the system robust and

enhance the performance of the Speaker Recognition System.

### Voice Activity Detection

Voice Activity Detection (VAD) [34] commonly referred as speech detection is the task of detecting voice segment from the given speech signal. The main goal here is to remove all silence segments from the speech signal and perform the desired operation on only speech segments [29]. This is a very important step in any Speech/Speaker Recognition task. It works by first finding the frame with maximum energy in the entire utterance and then a speech detection threshold is set. It then compares each frame energy against this threshold and discards them if its below this threshold.

### Feature Normalization

Feature Normalization [48] is basically carried out to compensate the differences in the training and test data that arises due to the use of different channels. The simplest way to perform Feature Normalization is to compute the mean of each feature over the entire speech utterance (for every training and test utterance) and remove this mean from each frame [9]. Hence, whichever channel the training and test speech originates from, the mean of every frame in the speech utterance becomes zero. This eventually reduces the effect of channel on the signal which otherwise could have become a major reason for degradation in the recognition performance. Some of the other normalization techniques that has also been used in the literature are feature warping [40], relative spectral (RASTA) filtering [31] and feature mapping [44].

### Score Normalization

Score Normalization [48] is another form of normalization that is basically applied to Speaker Verification Systems. The main idea behind Score Normalization is to set a common verification threshold which is done by transforming the scores from different speakers in a similar range [10]. The Score Normalization is given by the following equation

$$s' = \frac{s - \mu_1}{\sigma_1} \tag{2.10}$$

where $s$ and $s$' are the original and normalized score respectively and $\mu_1$, $\sigma_1$ are the estimated mean and standard deviation of the impostor score distribution [34].

Different Score Normalization techniques have been applied to Speaker Verification Systems. Some of the commonly used ones are Zero Normalization (Z-norm) [43], Test Normalization (T-norm) [10] and Handset Dependent Score Normalization (H-norm) [26][43]. These methods are found to be quite effective in reducing the Speaker Verification error rates [43]. In Z-norm [43] a set of impostor utterances is scored against each potential target model. The parameters $\mu_1$, $\sigma_1$ are then estimated from the resulting impostor score distribution. This normalization is however carried out during speaker enrollment.

In T-norm [10] which is performed during testing, the test utterance is also scored against some pre-selected set of non-target models which have similar session characteristics to that of the actual target model. The normalization parameters are then derived from the resulting score distribution. H-norm [26] method combines the advantage of both the Z-norm and the T-norm. Here Z-norm is first performed followed by the T-norm. It has also been very effective in compensating the channel mismatch problem however it requires both the offline and online normalization [26].

## 2.5 Research Trends: Current and Future

Speaker Recognition is one of the active and hot area of research in Speech Technology. Research and Development in this field has been carried out for over more than five decades [27]. In the beginning (early 1960s), the recognition was performed through Aural and Spectrogram comparison, then Template Matching models were used [27]. Then Dynamic Time Warping methods came into practice followed by the Statistical Pattern Recognition techniques like GMM, HMM, Neural Networks and showed an improvement in the recognition performance. Currently i-Vectors [24] have become state of the art techniques for Speaker Recognition with a very good performance [34][24].

Initially the speech corpus used for doing research in Speaker Recognition were created under clean laboratory conditions and were limited to very few speakers [27]. Recognition performance on such data set were found to be remarkable. As development progressed in this field more larger speech corpus with more number of target speakers [21] were released. These corpus were no longer originated under clean conditions rather the corpus were now built using telephone speech (landline channel) that consist of background noise. This made the task more challenging and realistic.

The progress and advances in the technology is often evaluated by NIST [25] that provides a common structure and setup for evaluation of Speaker Recognition Systems being developed across the world. The technology has progressed very well over the last 10 years [27] and many companies have commercialized Speaker Recognition Systems, particularly Verification Systems to be used as a biometric to control access to information and other services [22].

The current state of the art systems are based on short time spectral features [27]. However there are many other sources of information in speech signal which has not been explored fully. There is a need to explore higher order information in speech signal like prosodic features and other high level features which could make system more robust and contribute in enhancing the performance [34]. So identifying the best feature from a speech signal and researching towards finding a robust real time recognition system would be an interesting challenge in the research community.

## 2.6 Tools for Research

Various open source tools has been made available for doing research in Speaker Recognition so that one need not have to do it from scratch. The most popular and commonly used one is ALIZE toolkit [35] that is developed at Universite d' Avignon, France. Hidden Markov Model Toolkit (HTK)[2] is another popular statistical modeling package that is often used to extract features from the speech signal. Other tools like SVMLite [6], and SVMTorch [7] are available for implementing Support Vector Machines. For plotting the detection error tradeoff (DET) curves DETware toolbox provided by NIST [3] can be very useful. In this project ALIZE and HTK has been used for building GMM-UBM baseline system and extracting mfcc features respectively.

## 2.7 Summary

In this chapter we reviewed the literature on Speaker Recognition and covered various modeling paradigms that were used so far in the field. Different types of spectral features were also discussed that were used to build speaker models. Further some of the issues related to performance of the recognition system in terms of robustness were also briefly discussed.

# Chapter 3

# Kernel Machines for Speaker Classification

In this chapter we shall discuss the methodology based on Polynomial Kernels [15][8] for Speaker Identification. The input feature space is first transformed into a very high dimensional Kernel Space through polynomial expansion and a decision boundary (hyperplane) is placed between speaker and the impostor data in such a way that the number of misclassified data vector is minimized. This is the main idea behind training the classifiers using Kernel Machines which we shall discuss in this chapter.

## 3.1 Introduction : Polynomial Classifier

Traditional Statistical Classifiers like GMM creates probabilistic speaker model by estimating the mean and covariance of the training data [42]. Polynomial Classifiers on the other hand is a discriminatively trained classifier that requires feature vectors from both the target speaker and the impostor for building a speaker model.

### 3.1.1 Training the Classifier

The training of the classifier is based on minimizing the Mean Squared Error Criterion as discussed in [15]. Initially all the feature vectors $x_1, x_2....x_N$ are projected into higher dimensional Kernel Space using polynomial expansion function [16]. The classifier discriminant function is defined as

$$f(x) = \mathbf{w}^t \mathbf{p}(x) \tag{3.1}$$

where $\mathbf{p}(x)$ is the expanded super vector of polynomial basis terms and $\mathbf{w}$ is a vector of model parameters. For example second order polynomial expansion of a feature vector with two dimensional features $x = [x_1, x_2]$ yields six dimensional features [16] as shown below.

$$\mathbf{p}(x) = [1 \; x_1 \; x_2 \; x_1^2 \; x_2^2 \; x_1 x_2] \tag{3.2}$$

As the classifier shown in Figure 3.1 is a discriminative binary classifier, we have used one vs all training approach and trained 50 classifiers, one for each target speaker. Let $\mathbf{w}$ represent the desired class (model) and $y(w)$ be the ideal output with a value of 1 if $\mathbf{w}$ is a speaker and 0 for impostor. Therefore the resulting training equation can be stated as

$$\mathbf{w}_{spk} = \underset{\mathbf{w}}{argmin}\ \mathbf{E}\left\{(\mathbf{w}^t\mathbf{p}(x) - y(w))^2\right\} \tag{3.3}$$

where $\mathbf{p}$(x) represents the polynomial expansion of x and $\mathbf{E}$ represents the expectation over the data set X and the model $\mathbf{w}$. With target speaker data $x_1, x_2, ..., x_{N\,spk}$, and impostor data $y_1, y_2, ..., y_{N\,spk}$ above equation can be further simplified as

$$\mathbf{w}_{spk} = \underset{\mathbf{w}}{argmin}\ \left[\sum_{i=1}^{N_{spk}} \left|\mathbf{w}^t\mathbf{p}(x_i) - 1\right|^2 + \sum_{i=1}^{N_{imp}} \left|\mathbf{w}^t\mathbf{p}(y_i)\right|^2\right] \tag{3.4}$$

The above mathematical equation could also be expressed in a matrix notation as shown below. Here the target speaker and impostor data are represented by matrix $\mathbf{M}_{spk}$ and $\mathbf{M}_{imp}$ whose rows are the polynomial expansion [16] on the feature vectors of the respective training utterances. In other words, these matrices represents the speaker and impostor data in a higher dimensional kernel space.

$$\mathbf{M}_{spk} = \begin{bmatrix} \mathbf{p}(x_1) \\ \mathbf{p}(x_2) \\ . \\ . \\ \mathbf{p}(x_{N_{spk}}) \end{bmatrix} \ and\ \mathbf{M}_{imp} = \begin{bmatrix} \mathbf{p}(y_1) \\ \mathbf{p}(y_2) \\ . \\ . \\ \mathbf{p}(y_{N_{imp}}) \end{bmatrix} \tag{3.5}$$

Further the two matrices can also be combined and represented by a single matrix $\mathbf{M}$ as given below

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_{spk} \\ \mathbf{M}_{imp} \end{bmatrix} \tag{3.6}$$

Therefore using the above matrix notation, equation (3.4) can be written as

$$\mathbf{w}_{spk} = \underset{\mathbf{w}}{argmin}\ \|\mathbf{Mw} - \mathbf{o}\|_2 \tag{3.7}$$

where o denotes the vector of $N_{spk}$ ones and $N_{imp}$ zeros representing the ideal output for speaker and impostor [14]. Normal method equation [28] approach can be used to solve equation (3.7) as

$$\mathbf{M}^t\mathbf{Mw} = \mathbf{M}^t\mathbf{o} \tag{3.8}$$

The above equations are further simplified as

$$(\mathbf{M}_{spk}^t\mathbf{M}_{spk} + \mathbf{M}_{imp}^t\mathbf{M}_{imp})\mathbf{w}_{spk} = \mathbf{M}_{spk}^t 1 \tag{3.9}$$

Expressing these equations in terms of correlation matrix R gives us the following equation

$$\mathbf{Rw}_{spk} = \mathbf{M}_{spk}^t 1 \tag{3.10}$$

where

$$\mathbf{R} = \mathbf{R}_{spk} + \mathbf{R}_{imp} \tag{3.11}$$

$$\mathbf{R}_{spk} = \mathbf{M}_{spk}^{t}\mathbf{M}_{spk} \tag{3.12}$$

$$\mathbf{R}_{imp} = \mathbf{M}_{imp}^{t}\mathbf{M}_{imp} \tag{3.13}$$

and 1 represents the vector of all ones that signifies the output values for the target speaker being trained.

### 3.1.2 Classification/Recognition

Polynomial Kernel based classifier structure for performing Speaker Recognition is shown in figure 3.1.

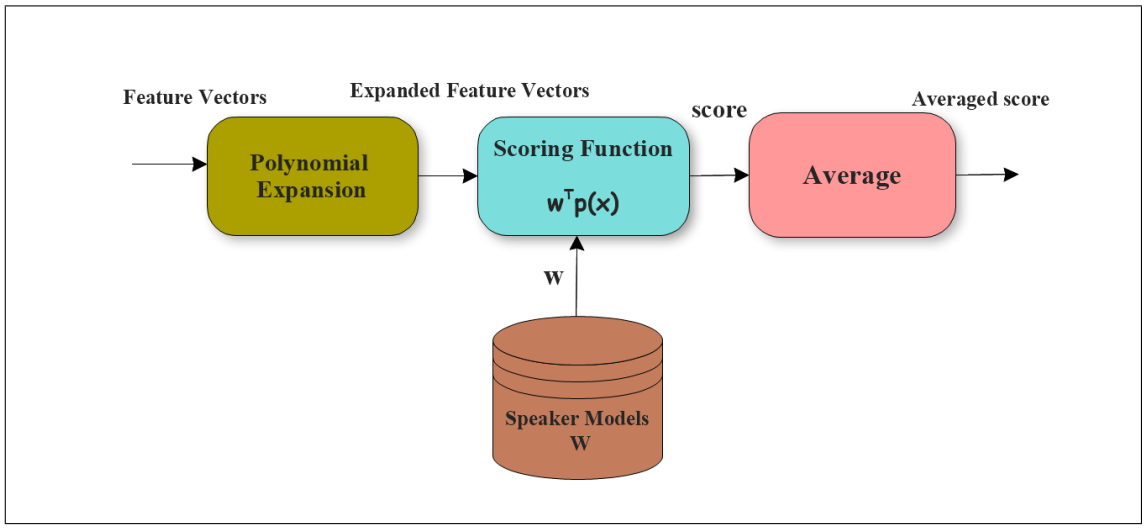


**Figure 3.1:** Structure of a Polynomial Classifier [8].

As can be seen there is a model $\mathbf{w}$ for each speaker. Each input feature vector $x_i$ is first transformed into the Kernel Space through polynomial expansion and a score is computed using the scoring function which is just a dot product between the model parameters and the expanded feature vector $\mathbf{p}(x_i)$. This dot product in other words is a measure of similarity between the model and the unknown test data. The final utterance score is generated by taking the average over the length of the utterance. Total score is given by

$$TotalScore = \frac{1}{N}\sum_{i=1}^{N}\mathbf{w}^{t}\mathbf{p}(x_i) \tag{3.14}$$

Here basically the score is normalized since the test utterances are of variable length.

With the above approach of computing per frame score and then finding the average score, the computational time incurred is very high. Therefore we have used a fast scoring method [16] which allows us to write the equation (3.14) in the following form

$$TotalScore = \mathbf{w}^t \left( \frac{1}{N} \sum_{i=1}^{N} \mathbf{p}(x_i) \right) \tag{3.15}$$

and compute the average over all the expanded vectors first and then find the score through a single dot product.

Hence the recognition operation using this structure works as follows: For verification task, a score is computed using the claimed speaker model for the unknown utterance. It is accepted as a valid speaker if the score is above the defined threshold else it is rejected as an impostor. However in case of identification, the score is computed using all the enrolled models and the one that yields the highest score is selected as a predicted model.

## 3.2 Major Steps

In this section we list out and discuss the major steps that were followed to implement the Speaker Identification System in the project.

### 3.2.1 Feature Extraction

Feature Extraction is the front end of any Speaker Recognition System. In this project we have used the Mel Frequency Cepstral Coefficients (MFCC) [23] features to build the speaker models. These coefficients are derived by taking the discrete cosine transformation on the melfilter bank energies $m_i$ as shown below

$$c_n = \sqrt{\frac{2}{P}} \sum_{i=1}^{P} m_i cos \left[ \frac{n(i - \frac{1}{2})\pi}{P} \right] \tag{3.16}$$

where P is the number of filterbank channels [23].

The speech data originates from the subset of NIST SRE-2008 data set [3]. More detailed description about the database is given in section 4.1. HTK toolkit [2] has been used for extracting the desired MFCC features. We have only used 12 static MFCC features in our entire experiment. Following command is used to retrieve the features using HCopy.

```
HCopy -C config input_speech.wav ouput.mfc
```

In the above command **config** is a standard configuration file that is used to define what features to extract from the input speech file. As we are considering only telephone speech in our project we need to extract MFCC features for both channels (A and B) for each audio file as there are two speakers involved in a telephonic conversation across the channels. Hence we have two configuration files **featExtract_A.cfg** and **featExtract_B.cfg** for channel A and channel B respectively.

The variable TARGETKIND is used to specify the type of features that we want to extract. If we want to have static and delta coefficients with energy then we need to set

this variable with `MFCC_E_D`. However in this project this variable is set to `MFCC_Z` as we are interested in only static coefficients with mean normalization. The term `'Z'` is appended to instruct the HTK to perform Cepstral Mean Normalization (CMN) along with feature extraction. Also we set the variable NUMCEPS to 12 to indicate the desired number of cepstral coefficients. Also we need to specify the channel information by setting the variable STEREOMODE as LEFT or RIGHT for left and right channels respectively.

HList is another function available in HTK that allows us to display the content of MFCC features on the command line which can be useful to check if the features have been extracted correctly or not.

### 3.2.2 Normalization and Voice Activity Detection (VAD)

Normalization [48] is performed by calculating the cepstral mean over whole utterance and removing this mean from each frame. The relevant parameters in the HTK configuration file as discussed in section 3.2.1 are set to perform the Cepstral Mean Normalization while doing feature extraction using the HCopy command.

Further, Voice Activity Detection [34] is performed to remove all the silence frames from the utterance, so as to ensure that the modeling of speaker is performed on voiced segments only while avoiding inclusion of silence/noise frames in the model. If the silence frames are not removed then the model being trained would include not only spectral characteristics of the speaker but it would include noise and silence also. This means that the degree of misclassification increases dramatically and the accuracy of the recognition becomes very poor. Hence this is one of the very important step in Speech/Speaker Recognition.

### 3.2.3 Training Impostors

The main advantage of this system [8] is the flexibility to train the impostor model (impostor matrix) separately and store it so that it could be simply loaded on to the training program while training the target speakers. The pseudo code for training the impostors is shown in Algorithm 1. As can be seen, three input parameters are required $flist$, $order$ and $impostorPath$. $flist$ is the file list that contains the path information of the MFCC features of the impostor files, $order$ denotes the order of polynomial to be used during expansion and the $impostorPath$ specifies the path where the impostor matrix is required to be stored.

Here the feature vectors are extracted from each impostor and silence frames are removed through VAD. Then polynomial expansion is performed using the polynomial order (provided as an input) to transform all the feature vectors into a Kernel space. These high dimensional frames are all accumulated row wise in a matrix $M_{imp}$. Then the Correlation Matrix $R_{imp}$ is computed as a product between $M_{imp}^t$ (transpose of $M_{imp}$) and the matrix $M_{imp}$ that basically measures the similarity between all the newly generated

**Algorithm 1** Following algorithm is used to create an Impostor Matrix

1: **procedure** TRAINIMPOSTOR($flist, order, impostorPath$)
2: Initialize $R_{imp}$ as an empty structure to store impostor matrix.
3: **for each impostor file in *flist* do**
4: *data* ← *retrieve feature vectors for impostor file* ▷ Feature Extraction
5: *newData* ← *VAD(data)* ▷ Voice Activity Detection
6: *allRows* ← *expandFrame(newData, order)* ▷ Polynomial Expansion
7: *append allRows in matrix* $M_{imp}$
8: $M_{imp}^t \leftarrow tranpose(M_{imp})$ ▷ Find transpose
9: $R_{imp} \leftarrow mulitply(M_{imp}^t, M_{imp})$ ▷ Impostor Correlation Matrix
10: Save the impostor matrix $R_{imp}$ permanently in $impostorPath$.

cross correlated features in the higher dimension. Thus we find a very high correlation in the diagonal elements of the Correlation Matrix in comparison to non diagonal elements.

### 3.2.4 Training Target Speakers

The pseudo code shown in the Algorithm 2 is followed to train all the target speakers. Initially the impostor matrix is loaded into the program and for each speaker file, feature vectors are extracted followed by Voice Activity Detection that is being performed to remove all the silence frames.

Polynomial expansion is then performed to convert the MFCC features into higher dimension, which is stored in a speaker matrix $Mspk$. Its transpose is computed next. Then the Correlation Matrix is computed as shown in step 9 of the algorithm 2. The speaker and impostor Correlation Matrix is combined together to get a final Correlation Matrix $R$ as shown in step 11. The vector $A_{spk}$ in step 10 signifies the particular speaker being modeled. Computation of the required model parameters $\mathbf{w}$ for the speaker is done by performing Cholesky Decomposition which is much efficient computationally in comparison to finding inverse of the Correlation Matrix and multiplying with the speaker vector $A_{spk}$.

Cholesky Decomposition [1] works as follows: First it decomposes the Correlation Matrix $R$ into lower triangular matrix $L$ and its transpose $L^t$. Then it solves the linear system of equation for $Y$ once and uses this solution to solve another linear system for model $\mathbf{w}$ as shown in steps from 12 to 15 in the algorithm. Next model id of the speaker file is extracted from the reference list that was supplied with the data set. The model id and the respective model parameters are appended in the data structure MODELS which in our case is a python list.

In this way model id and model parameters $\mathbf{w}$ for each speaker file is computed and appended in the MODELS which is then saved permanently in the given path information. The training algorithm discussed here has been used from [16].

**Algorithm 2** Following algorithm is used to train Target Speakers

1: **procedure** TRAINSPEAKER($flist, order, impostor, modelFile$)
2: Initialize MODELS as an empty structure to store trained models.
3: $Rimp \leftarrow impostor$ ▷ Load Impostor Correlation Matrix
4: **for** `each speaker file in` *flist* **do**
5: $data \leftarrow$ *retrieve feature vectors for speaker file*
6: $newData \leftarrow$ *VAD(data)* ▷ Voice Activity Detection
7: $M_{spk} \leftarrow$ *expandFrame(newData, order)* ▷ Polynomial Expansion
8: $M_{spk}^t \leftarrow tranpose(M_{spk})$
9: $R_{spk} \leftarrow mulitply(M_{spk}^t, M_{spk})$ ▷ Speaker Correlation Matrix
10: $A_{spk} \leftarrow mulitply(M_{spk}^t, 1)$ ▷ 1 is a Vector of Ones
11: $R \leftarrow add(R_{spk}, R_{imp})$ ▷ Final Correlation Matrix
12: Solve $RW_{spk} = M_{spk}^t 1$ for target speaker model $W_{spk}$
13: $L \leftarrow cholesky(R)$ ▷ Using Cholesky Decomposition
14: $L^t \leftarrow transpose(L)$ ▷ Transpose Lower Triangular Matrix L
15: $Y \leftarrow solve(L, A_{spk})$ ▷ Solve Linear System LY = Aspk
16: $W_{spk} \leftarrow solve(L^t, Y)$ ▷ Solve for Speaker Model
17: $modelID \leftarrow getSpeakerId(speaker)$
18: Append model parameters $W_{spk}$ and $modelId$ in MODELS.
19: Save MODELS in the the file $modelFile$.

### 3.2.5 Test Models

To test the performance of the speaker models on the unknown test utterance the pseudo code shown in Algorithm 3 is used. As can be seen, various parameters that are passed to the algorithm are: list of test files *flist*, order of expansion *order*, list of trained models *modelList* and path *resultPath*, used to store the result of classification on the test utterances.

For each test utterance, feature extraction is first performed to get 12 dimensional MFCC features followed by Voice Activity Detection (VAD) that eliminates the silence frames from the utterance as shown in step 6. Then these input features are transformed into a high dimensional kernel space through polynomial expansion as shown in step 7 of Algorithm 3.

Step 9 through 12 computes the score for each test utterance using all the enrolled speaker models in the *modelList* and picks one for which score is maximum as the predicted speaker of the unknown test utterance.

### 3.2.6 Scoring and Evaluation

In this section we outline the algorithm used for computing the model score and evaluating the accuracy of the Identification System. The steps shown in Algorithm 4 is used to

**Algorithm 3** Following algorithm is used for Testing Speaker Models

**procedure** TESTMODELS($flist, order, modelList, resultPath$)
  $predictionList \leftarrow$ *empty list data structure*
  $models \leftarrow modelList$ ▷ Load the Models
  **for** `each test file in` $flist$ **do**
    $data \leftarrow$ *retrieve feature vectors for test file*
    $newData \leftarrow$ *VAD(data)* ▷ Voice Activity Detection
    $speechFrames \leftarrow$ *expandFrame(newData, order)* ▷ Poly Expansion
    $scoreList \leftarrow$ *empty list data structure*
    **for** `each model w in` $modelList$ **do**
      $modelScore \leftarrow computeScore(\mathbf{w}, speechFrames)$ ▷ Score using model w
      *append modelScore in scoreList*
    $predictedSpeaker \leftarrow max(scoreList)$ ▷ Speaker with max Score
    *append predictedSpeaker in predictionList*
  Save $predictionList$ permanently in $resultPath$.

compute per frame score and find the average over the length of the test utterance. However this method of scoring involves a huge computation time as it has to compute score for each frame. Therefore we have used a fast scoring method as discussed in Section 3.1.2 to efficiently compute the score over the entire test utterance by taking just a simple dot product that measures the similarity between the speaker model and the unknown test utterance.

**Algorithm 4** Computing the score using the Speaker Model

**procedure** COMPUTESCORE($\mathbf{w}, speechFrames$)
  $totalScore \leftarrow 0$
  **for** `each vector x in` $speechFrames$ **do**
    $score \leftarrow \mathbf{w}^t\mathbf{x}$
    $totalScore \leftarrow totalScore + score$
  $averageScore \leftarrow totalScore/length(speechFrames)$
  return $averageScore$

The identification accuracy is defined as the number of correctly classified test utterances over the total number of utterances being tested [46]. The pseudo code shown is Algorithm 5 is used for this purpose. The *referenceList* is passed to this algorithm which is actually a database that contains information about the list of enrolled speakers along with the identifiers of their test utterances. The *predictionList* that consist of the predicted speaker model for each test utterance is then compared with the *referenceList* as shown in Algorithm 5 and the identification accuracy is computed.

**Algorithm 5** To Compute the Identification Accuracy

```
1: procedure COMPUTEACCURACY(predictionList, referenceList, segmentCount)
2:     totalHit ← 0
3:     for i := 1 to segmentCount do
4:         if predictionList[i] = referenceList[i] then
5:             totalHit ← totalHit + 1
6:     accuracy ← (totalHit/segmentCount) * 100
7:     return accuracy
```

## 3.3 Concatenation of Frames: Proposed Approach

The proposed method requires chosing the window size, which is basically the number of frames to be concatenated (from left and right) to the current frame being taken into consideration. Let us consider the window size chosen to be N and original dimension of the frame as D.

This method is computationally very challenging and expensive because of the massive memory requirements as we increase the window size and the order of polynomial. However incurring such a huge computational effort, we expect our new proposed system to perform better with more accuracy over the traditional approach where we were just expanding a single frame but in this approach we are incorporating more information about the speaker by concatenating frames in time domain.

### 3.3.1 Method A: Feature Concatenation and Expansion

In this method we basically concatenate the first dimensional feature across N frames and expand them using polynomial expansion to get a new higher dimensional vector as the first dimensional feature of the new frame. This process is repeated for remaining D-1 dimensional features. Once this process is completed, we get a very high dimensional super vector as a resulting frame from this approach. This procedure is repeated for each frames in the training/test utterances.

The proposed concept is shown in Figure 3.2. All other steps during training and testing remains the same as in the traditional approach we discussed in Section 3.2.4 and 3.2.5. The only difference is that before expanding the frames we perform concatenation of features across all the N frames dimension wise (for all 12 dimension in original frame) to get a new frame and then polynomial expansion is performed.

### 3.3.2 Method B: Frame Concatenation and Expansion

Here the idea is to simply concatenate the N (window) frames and expand them into higher dimensional Kernel Space through polynomial expansion. The only drawback here is the high computational cost incurred for doing the experiments. The memory requirement becomes massive for even 5 frame concatenation with order 3 expansion. The new

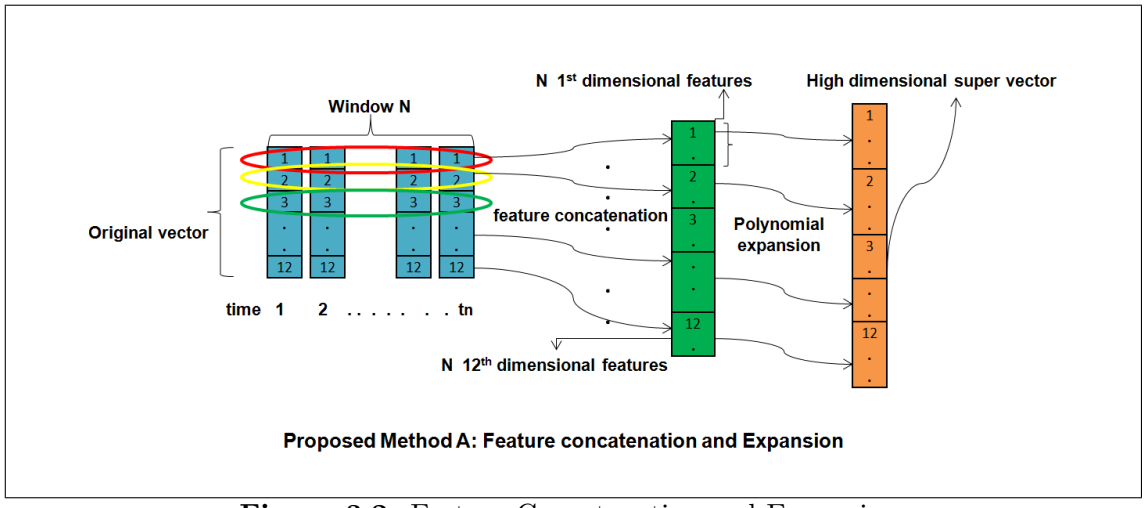


**Figure 3.2:** Feature Concatenation and Expansion.

dimension of the concatenated frame becomes 60 (original frame is 12 dimensional) and **order 3** expansion of this 60 dimensional frame yields **39711** dimensional super vector. Each frame will be of such a huge dimension, and impostor matrix computation involves lots of feature vectors from general population which makes this process computationally very challenging. Due to this reason we have only implemented order 3 system with 3 frame window. The concept discussed under this new method can be seen in Figure 3.3.

This was infact one of the key reason for coming up with the idea of feature concatenation across N (window) frames first and expanding them as discussed in previous method A.

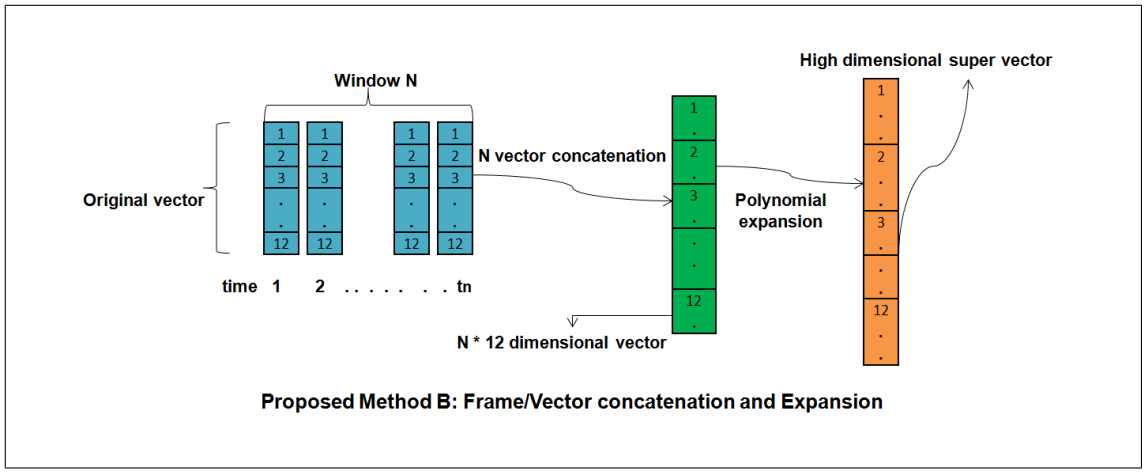


**Figure 3.3:** Frame Concatenation and Expansion.

### 3.3.3 Method C: Frame Concatenation without Expansion

This approach is a very simple approach based on just concatenation of frames in the input feature space. As shown in figure 3.4 we concatenate N (window) frames first and obtain a new high dimensional concatenated frame. However we do not perform any

polynomial expansion which means in our new frame we do not have any cross correlated polynomial basis terms. The new frame consist of simple concatenation of features in the input space, thus all the computation in this method would be actually carried out in the input space itself. All other steps shall remain same as described in the previous Sections 3.2.4 and 3.2.5 except that now we do not perform polynomial expansion.

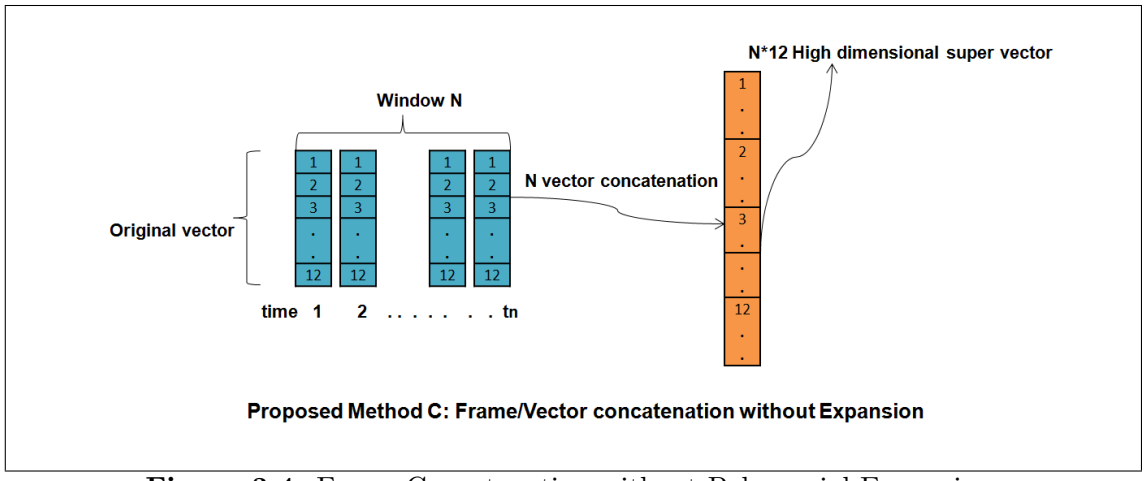


**Figure 3.4:** Frame Concatenation without Polynomial Expansion.

## 3.4 Summary

In this chapter we have discussed the methodology for building Speaker Identification System based on transformation of input feature space into a high dimensional Kernel Space through Polynomial Expansion. Various steps needed to perform the task were discussed in detail and pseudo codes were also provided for different tasks involving training and testing the target speaker models. Further improvement over the existing methodology were proposed that basically involves concatenation of frames in the time domain in order to include more wider temporal context to improve the performance of the Speaker Identification System.

# Chapter 4

# Baseline Setup

## 4.1 Database Description

The **database** used in this project is a small subset "Part1 drive2 (p1d2)" of the **NIST SRE-2008** evaluation corpus [3]. It consist of 985 stereo files. These stereo files were further splitted into two channels: channel A (left) and channel B (right) to obtain a total of 1970 single channel audio files. The total duration of this database is 152 hours 40 minutes. The corpus basically comprises of two types of speech: telephone speech and interview speech. About two third of the database **p1d2** are interview speech and one third are of telephonic speech type. The interview speech files are of approximately 3 minutes duration while telephonic speech files are of about 5 minutes duration. However in this project, we have only used the audio files of type telephone speech.

There are 210 unique speakers under telephone speech category in this database (**p1d2**). We prepared a file listing all speakers along with their test utterances that are labeled 'target'. We discarded all the utterances that were labeled 'nontarget' because we are primarily concerned in closed set Identification. So we want to collect those speakers for which there are sufficient test utterances where the target speaker is speaking. We then picked 50 speakers which had atleast 3 test utterances as our working data set to do the experiments.

For training impostors we used audio files that were not used during training and testing the target speakers. However one important thing to note here is that a speaker can have more than one model for which audio file segment originates under different condition. We have carefully selected the speaker models in such a way that the model id (for the associated training segment) belongs to only one unique speaker. This means that the 50 targets we have selected are trained using unique training segments from the database.

The following are the key files provided by the NIST [3] that provides all necessary information about the speech data available in the corpus which are being used to prepare various training and test list during the experiment. A sample content of each file is provided to show the different information provided by these files.

- **model.key :** Each line in the file contains the following details.

  ```
  10017,f,tdvlv:a,110118,interview,mic-12,ENG,USE
  ```

  It gives information about (left to right) the model id, gender, segment id with channel, speaker id, type of speech (phonecall or telephonic), type of channel (microphone channel or main phone line), language and speaker native language. This file is basically used to extract the model id associated with the particular training segment id.

- **train.segment :** Each line gives the following information.

  ```
  taacu,22155,short2,phonecall,CHN,109659:phn-main:USE
  ```

  It specifies (from left to right) the segment id of the training data, ldc id a unique identifier in the ldc corpus that is given to a specific conversation and recording per speaker, condition (whether short2 or short3), type of the speech (phonecall or interview speech), language, and speaker id with channel information.

- **trial.key :** Each line provides the following information.

  ```
  10017,fivsw,b,nontarget,N,N,N,Y,N,N,N,N
  ```

  where the first 4 parameters are model id, test segment id, channel and trial status respectively while the remaining 8 parameters specify different evaluation conditions [3].

### 4.1.1 Data Definition

The Data definition section is included here to give various data files a unique name to enhance the clarity in understanding its meaning and purpose in the experiment.

- **p1d2**: is the database that we have used in our experiment. It is a subset of the NIST SRE-2008 corpus as discussed in section 4.1.

- **trainData50**: this is a data set consisting of 50 speakers training data taken from the **p1d2** database. This was the data set used for performing our first experiment in the project.

- **test42**: is a test data set consisting of 42 utterances from 42 target speakers used to test the initial system.

- **trainData210**: is the total set of 210 speakers under telephonic speech category in the **p1d2** database. It is used to train a population of 210 target speakers.

- **newTrainData50**: is a data set of 50 training utterances from 50 speakers chosen in such a way that there exist atleast three test utterances in **p1d2** labelled as target. In other words this training data set was chosen ensuring that we have 3 test utterances per speaker where the speaker is actually speaking .
- **trainData48**: is a set of 48 target training utterances where we have atleast one minute training utterance after performing Voice Activity Detection (VAD).
- **impostor500**: is an impostor list of 500 training utterances that is taken from **p1d2** which is completely independent of the training and test data set.
- **impostor86**: is a list of 86 audio files that is independent of training and test data set used for training impostor matrix.
- **impostor49**: is a list of 49 audio files (from the training data set **newTrainData50**) used to train the impostor matrix. There will be 50 **impostor49** data set, one for each target speaker.
- **test1024**: consist of the entire test utterances labeled as target in the **p1d2** database. This was used to test 210 target speaker models trained using **trainData210**.
- **test150**: is our main test data set consisting of 150 test utterances where target speaker is speaking. This data set is designed to test the models trained using **newTrainData50**.
- **test144**: is the test data set designed for 48 speakers in **trainData48** set with 3 test utterances (trials) per speaker.

### 4.1.2 ALIZE Configuration Files

The following are various configuration files of the ALIZE tutorial that were used for building the GMM-UBM models.

- **featExtract_A.cfg**: Channel A configuration file used for extracting MFCC features using HTK.
- **featExtract_B.cfg**: Channel B configuration file used for extracting MFCC features using HTK.
- **norm_energy.cfg**: Configuration file used for performing Energy Normalization.
- **energydetector.cfg**: Configuration file used for performing Voice Activity Detection.
- **normFeatures.cfg**: Configuration file used for performing Feature Normalization.

- **trainWorld.cfg**: Configuration file used for training the Universal Background Model.
- **trainTarget.cfg**: Configuration file used for training the target speaker by performing MAP adaptation on UBM.

## 4.2 GMM-UBM Training

The baseline GMM-UBM system is built using the ALIZE toolkit [35]. It provides a step by step guideline for generating the baseline system . The parameters in the configuration files for various operations like feature extraction, energy normalization etc. were set appropriately as per the requirement. These file comes as a part of the ALIZE tutorial, hence we do not require to create it on our own from scratch. It just requires us to make very few changes in some parameter values while most of the parameters are kept same as they were before (default values). We discuss these in the following section.

### 4.2.1 Feature Extraction

ALIZE provides two different options for reading the MFCC features from the raw speech file. The first one is using the SPRO tool [5] and the other is with the HTK toolkit [2] using HCopy command. We have used the HTK option in our work. The configuration files **featExtract_A.cfg** and **featExtract_B.cfg** for both the channels were set appropriately and 12 dimensional MFCC features were extracted as discussed in Section 3.2.1.

### 4.2.2 Energy Normalization

This step is carried out to normalize the energy component in each frame of the speech utterance so that Voice Activity Detection can perform better in distinguishing speech from the silence/noise frames. It works by first computing the mean and variance of the energy (13th coefficient in the MFCC vector) over the entire utterance. This mean is then removed from each frame energy and divided by the variance in order to make each frame energy have zero mean and unit variance.

The parameter *loadFeatureFileVectSize* in the configuration file **norm_energy.cfg** is initialized '13' (instead of 60 which is the default value used by ALIZE). The 13th coefficient in the MFCC vector is the energy of the frame. Also the parameter *featureServerMask* is set to '13' (default is 19) that instructs the system to just select the energy component (13th coefficient) for normalization. Further the input file format specifier *loadFeatureFileExtraction* is set to '.mfc', and the output file format specifier *saveFeatureFileExtraction* is set to '.nrg.mfc',.

### 4.2.3 Voice Activity Detection

The next important task to be carried out is removal of silence frames from the speech utterance in order to enhance the system performance by ensuring that our models are trained using only voiced speech segments. Hence we perform Voice Activity Detection (VAD) for this purpose. All the parameters in the configuration file **energydetector.cfg** are set to the default values except the parameter *loadFeatureFileExtension* which is set to '.nrg.mfc' since we specified '.nrg.mfc' as output file extension while doing energy normalization. Using this configuration file the VAD is carried out to remove all the silence frames from the utterance.

### 4.2.4 Feature Normalization

The audio files present in the database **p1d2** originates from different channels, therefore while building background and target models it is very important to eliminate the influence of channel variation in the model. Hence Feature Normalization is often performed to suppress these channel effects as discussed in Section 2.4.5.

The parameters *saveFeatureFileExtension* is changed to '.norm.prm', *vectSize* to '13' (default is 60) and *loadFeatureFileExtension* to 'norm.feat.mfc' in the configuration file **normFeatures.cfg**. Other parameters were set to their default values.

### 4.2.5 UBM Training

The Universal Background Model (UBM) is trained using **impostor500** that consist of 500 telephonic speech utterances from the **p1d2** database that were not used during training or testing the target models. This is a speaker independent general model trained to capture the general characteristics of a large number of speaker population as discussed in Section 2.3.3. The energy coefficient is removed from each frame. The number of gaussian component that has been chosen to build the UBM is 2048 and the number of (EM) iterations for estimating the optimal model parameters is 8. These parameter values were set appropriately in the configuration file **trainWorld.cfg** and the background model is built using the configuration file.

### 4.2.6 Training Target Speaker

Front end processing on each target speaker utterance is performed as described in the previous sections. Then speaker enrollment/training is performed through maximum a posteriori (MAP) adaptation on the background model (UBM) with the speaker data to make it speaker specific model representing a particular speaker.

The energy coefficient is removed from each frame and MAP adaptation method is selected as the underlying adaptation algorithm in the configuration file **trainTarget.cfg**. We also set the parameter *meanAdapt* to 'true' to specify that we only want to perform the

mean adaptation. Likewise many other relevant parameters such as file paths are properly initialized and the target models are trained using the configuration file **trainTarget.cfg**.

## 4.3 Testing

Identification of an unknown test utterance is now performed by computing the score using the MAP adapted speaker model and the background UBM model. Then the log likelihood difference between the target score and the background score is computed. This becomes the final score for the given test utterance. The procedure is repeated for all the enrolled speaker models in the system. Then the GMM-UBM Identification System selects the model for which score is highest as the predicted speaker of the unknown test utterance as shown in Figure 2.2.

## 4.4 Summary

ALIZE toolkit that provides a step by step guideline, was used for building the GMM-UBM baseline system. UBM was trained using 500 telephonic speech data with 2048 gaussian mixture components. 50 speakers in the data set **newTrainData50** were trained by performing MAP adaptation on the background UBM model to create Speaker Dependent GMM-UBM models. Testing were then performed on **test150** data set consisting of 150 test utterances. Various steps involving feature extraction, normalization etc are carried out as discussed in the chapter.

# Chapter 5

# Experiments and Results

In this chapter we shall describe various experiments that were performed during the project.

## 5.1 Baseline Results

The result of the GMM-UBM baseline system using 12 dimensional static MFCC features is shown in the following table.

| **Baseline System** | **Identification Accuracy %** |
|---|---|
| GMM-UBM | 57.3 |

Table 5.1: Baseline GMM-UBM Results

The results shown above is obtained by testing the system on data set **test150** consisting of 150 test utterances. The result is not as good as we expected, the primary reason could be because of the very fact that the data set is not a clean speech. It is a noisy telephone speech data that contains lots of silence and filler sound (like umm , uhhh, mmmm).

## 5.2 Initial Experiments

As our first task in the project was to replicate the system as described in [8][16] and then extend it with concatenation of neighboring frames we decided to run our initial experiments with a small data set and trained 50 speakers using **trainData50** and tested the performance on **test42**. The recognition result is shown in table 5.2.

We observe that the system performed well on training data and gave **100%** identification accuracy with polynomial order 3 and 4. The performance on unseen data seems to improve with increase in the polynomial order. Though we did not get a remarkable performance on test set, however it gave an intuition that the methodology and our approach was working correctly. With this experimental observation we decided to run

| Polynomial Order | Training Data % | Test Data % |
|---|---|---|
| 1 | 42.0 | 14.6 |
| 2 | 94.0 | 39.0 |
| 3 | 100.0 | 58.5 |
| 4 | 100.0 | 63.4 |

Table 5.2: Performance of the system for different polynomial order that is trained using **trainData50** and **impostor86** data set while tested on **test42** test set.

experiment on larger speaker population with sufficient test utterances for each speaker model.

Therefore we used entire data set **trainData210** to train 210 target speakers. Testing was further performed using the entire telephonic test set **test1024**. Following table shows the performance of the system.

| # Impostors | Polynomial Order | Identification Accuracy % |
|---|---|---|
| 86 | 2 | 7.3 |
| | 3 | 11.9 |
| 500 | 2 | 8.0 |
| | 3 | 12.9 |

Table 5.3: Performance of the system trained using **trainData210** with **impostor86** and **impostor500** while testing being performed on **test1024** data set

As we increase the polynomial order, the number of basis terms (cross-correlated terms) increases during expansion into higher dimension, therefore we see an increase upto **4%** accuracy in order 3 compared to order 2 results. Also we find that using more number of impostors leads to some improvement in classification performance. With 500 impostors we observe a gain of about **0.7%** with order 2 and **1%** with order 3 polynomial expansion.

However the results are very poor and not upto an acceptable figure. We found two reasons for this. Firstly, its because of the fact that with large target population number of comparisons increases while increasing the probability of false classification. Hence there is a high chance for degradation of overall performance. Secondly, we found that the test data was not uniformly distributed among the 210 speaker in the **test1024** data set. There were some speaker without any test utterances and some with very less test utterances (1 to 4). However for few target speakers there were many test utterances (20 to 40). So this imbalance in the test utterance per speaker creates a bias in overall system performance.

This made us think to chose our training set and test set more carefully from the database **p1d2** with uniform test utterances for each target speaker being modeled. We then chose 50 target speakers training data set **newTrainData50** in such a way that

each speaker in this set had 3 test utterances where these speakers were found speaking. The **test150** is the desired test data set consisting of 150 test utterance (3 per target speaker) which is used to perform our further experiments.

## 5.3 Different Impostor Population

In this experiment we have used **newTrainData50** to train 50 speakers and tested using **test150** data set. The motive here is to analyze more closely the impact on identification performance with different impostor population. In first case we trained each speaker against 49 speakers. In other words while training classifier for a particular speaker remaining 49 speakers were used as impostor. The process was repeated for all speakers. In second case we used **impostor500** data set as impostor and trained the classifier for each speaker against this impostor using one vs all binary classification.

Since the number of impostors is relatively larger than the speaker so we used only 30 seconds speech data from each impostor audio file to build the impostor model (i.e the impostor Correlation Matrix). With 30 seconds speech most of the general characteristics of the general speaker population can be captured while the impostor matrix dimension can be kept to an optimum number.

| # Impostors | Polynomial Order | Identification Accuracy % |
|---|---|---|
| 49 | 2 | 19.3 |
| | 3 | 26.0 |
| | 4 | 28.0 |
| 500 | 2 | 21.3 |
| | 3 | 26.0 |
| | 4 | 30.6 |

Table 5.4: Performance on **test150** when target models are trained using different number of impostors

As can be seen while using an unseen data set of 500 impostors **impostor500** there is an improvement of about **2%** over **impostor49** set that consist of 49 speakers from the training set. As we increase the polynomial order from 2 to 4 we see a consistent raise of about **4%** to **5%** accuracy with **impostor500**. However with **impostor49** the improvement with increase in polynomial order is not so consistent as can be seen from the table. There is a gain of **6.67%** with order 3 compared to order 2 results, however with order 4 its just **2%** gain in comparison to order 3 results. This shows that the system gives good performance when we use unseen data as impostors.

## 5.4 Compensation for Different Length Training Data

After doing VAD, we found that the training and test utterance for all the target speakers are never same. Though all training and test data are of 5 minutes length but it contains lots of fillers (like ummm, uhh,mmmm etc) and silence frames. The data are never uniformly distributed. This means that some speaker models are over trained with more data while some are under trained as there were very small amount of actual speech where the speaker was actually speaking some words that could contribute significant phonetic content which could be modeled properly. There were two speaker models in **newTrainData50** which were actually trained on just 23 sec and 43 sec of speech as remaining frames were all silence and noise which were removed during Voice Activity Detection. So these models would not be able to perform well during recognition in comparison to other models trained with good amount of input data.

Motivated from the above fact, we thought of experimenting the system with some compensation with variable length training data among different target speakers and observe differences if any. Though there are many compensation strategies [12] we have used a simple compensation method as discussed in [8][12]. The basic idea here is to divide the score of the classifier by the training data normalization factor. The new score is given as,

$$newScore = \frac{score}{norm_i} \tag{5.1}$$

where $norm_i = \frac{N_i}{N}$ is the speaker $i$ training data normalization factor, $N = \sum_{j=1}^{N_{spkr}} N_j$ is the total number of speech frames in the entire training set (all speakers) and $N_i$ is the number of frames in speaker i training data.

| # Impostors | Order | No Compensation % | With Compensation % |
|---|---|---|---|
| 500 | 2 | 21.3 | 22.0 |
| | 3 | 26.0 | 28.0 |
| | 4 | 30.6 | 32.0 |

Table 5.5: Performance of the System on **test150** test set with and without compensation on training length.

From the above results we see that with inclusion of above compensation method there is a gain of about 2% in the identification accuracy.

## 5.5 Varying Length of Training and Test Data

In this experiment our main objective is to see the impact in the identification performance when models are trained and tested on different length training and test utterance. We removed two target speakers having less than 60 seconds utterance (after VAD) from the list and created a new training set of 48 speakers called **trainData48** and evaluated the performance on **test144**.

### 5.5.1 Fixed Amount of Training Data

In this part of experiment we trained 48 target speakers using 30 seconds speech for order 2, 3 and 4 polynomials and tested the performance on test utterance of length 30 second, 15 second and 1 second respectively. The following table gives the identification accuracy for the different cases we considered.

| **Amount of Training data** | **Polynomial Order** | **Length of Test Segment** | | |
|---|---|---|---|---|
| | | **30sec %** | **15sec %** | **1sec %** |
| 30 seconds | 2 | 18.7 | 16.6 | 12.5 |
| | 3 | 25.0 | 20.1 | 13.2 |
| | 4 | 26.38 | 23.6 | 15.9 |

Table 5.6: Performance of the system trained using **trainData48** data set with 30 seconds speech per target speaker while testing is performed on **test144** with different test length.

We observe the following facts from the above experiment:

- The best performance was found for 30 second test segment among the three test length segments. This is true for all the polynomial order. It means that performance becomes better with good length of test data.
- As the size of test data decreases from 30 seconds to 1 seconds we observe a decrease in the performance as well.
- As we increase the polynomial order, performance of the recognition increases as can be seen in the above table.

### 5.5.2 Fixed Amount of Test Data

Here we trained the system with two different utterance length 30 seconds and 60 seconds respectively while testing was performed on 5 seconds utterance. We have chosen a very small test utterance length in order to analyze the system behaviour under limited test data. The following table gives the identification accuracy obtained under different training criterion.

| **Amount of Training data** | **Performance on 5 second speech %** | | |
|---|---|---|---|
| | **Order 2** | **Order 3** | **Order 4** |
| 30 seconds | 18.7 | 18.7 | 24.3 |
| 60 seconds | 18.1 | 20.1 | 25.0 |

Table 5.7: Performance of the Identification System trained using **trainData48** with different training utterance length while tested on **test144** test set of 5 seconds length each.

As can be seen from Table 5.7, best performance is achieved for order 4 polynomial in both the cases (i.e training with 30 seconds and 60 seconds data). The accuracy seems to be similar with order 2 and order 3 when system is trained on 30 seconds speech data. The reason could be because of the fact that the test utterance is very short and model has been trained with very less training data. However when the system is trained on 60 seconds data, we observe that performance increases as we increase the order of the polynomial.

This gives a very clear intuition about the fact that if a model is trained with larger set of training data then the system becomes more powerful and can perform reasonably well during recognition compared to the one being trained with lesser data.

Further from Table 5.6, we observe that with an increase in length of test utterance from 1 second to 30 second there is an increase in the recognition accuracy. Thus it gives an idea about the necessity of having a sufficiently good amount of training and test data to have a good identification performance.

## 5.6 Concatenation Results

This section describes the results of the various experiments that were performed using the proposed method of concatenating neighbor frames.

### 5.6.1 Method A

The following table 5.8 shows the performance of the system based on proposed Method A with order 2 and 3 polynomial expansion for different window size.

| # Impostors | Order | Window (N) | New Frame Size | Accuracy % |
|---|---|---|---|---|
| 500 | 2 | 19 | 2520 | 16.3 |
| 200 | 3 | 7 | 1440 | 18.0 |
| | | 11 | 4368 | 19.3 |

Table 5.8: Identification Performance of the System on **test150** data set for different polynomial order and window size. Original frame size is 12.

The result was not as good as we were expecting from our proposed system. The system did not perform well as compared to our initial system for both order 2 and order 3 polynomials. As can be observed from Table 5.4 the accuracy of the initial system with 500 impostor and order 2 polynomial is **21.3%** while the new system just gave **16.3%**. Also with order 3 polynomial the new system could give only **19.3%** which is very less in comparison to our initial system identification accuracy of **26.0%**.

The reason why our proposed system failed to give good results could be because of the fact that we are loosing cross correlation between the features in the higher dimensional kernel space as we had concatenated the features in N (window) frames dimension wise i.e

we took all the first dimension feature from all N frames and expanded these N features using polynomial expansion and again we did it for second dimension across all frames and so on. In the process when expansion takes place, we do not get any cross correlated polynomial basis terms in the kernel space. Due to this reason system performed poorly compared to the initial system.

### 5.6.2 Method B

In this method we concatenate the frames directly and perform the polynomial expansion as discussed in Section 3.3. The results shown in Table 5.9 gives a message that the conclusion we had drawn for the failure of method A was indeed correct as we see a very good improvement over the accuracy when concatenating frames directly. The reason is because Method B preserves the cross correlation between the features when transformed into higher dimensional space through polynomial expansion. Hence we observe improvement in the performance in comparison to the initial system.

For polynomial order 2 we ran two experiments. In the first we used 500 impostors with 3 frame concatenation window and got **25.3%** identification performance which is better than the initial system accuracy of **21.3%**. Next we tried reducing the impostor size to 100 while increasing the window size to 11. In this case we obtained more better accuracy of **32%**. This means that an increase in window size improves performance however it also increases the computational cost dramatically.

Next with polynomial order 3 we performed only one experiment because of the computational constraint. We used 100 impostor and 3 frame window in this case and achieved an accuracy of **32%** which is an improvement over the initial system accuracy of **28%** with polynomial order 3.

| **# Impostors** | **Order** | **Window (N)** | **New Frame Size** | **Accuracy %** |
|---|---|---|---|---|
| 500 | 2 | 3 | 703 | 25.3 |
| 100 | 2 | 11 | 8911 | 32.0 |
| 100 | 3 | 3 | 9139 | 32.0 |

Table 5.9: Performance of the System on **test150** that is trained using Frame Concatenation and Expansion approach. Original frame size is 12.

Though we are unable to perform experiments by concatenating large number of frames with higher order polynomials due to the high computational cost involved, but above results proved that our proposed algorithm works well and gives a good improvement over the initial system. With order 2 polynomial we got an improvement of more than **10%** when **11 frames** were used as concatenation window. Also with order 3 it showed an improvement of **4%** over initial system. It also showed that the accuracy with 11 frames and order 2 seems to be similar with that of order 3 system with 3 frame concatenation. However computationally order 3 is very expensive in comparison to order 2 system as

can be seen in the table, the number of features generated is **9139** per frame with order 3 polynomial expansion while its **8911** with order 2. But there is one point to be noticed that with small polynomial order we are able to increase the window size and observe an increased performance which are comparable (almost similar) and may be even better than what we could have achieved with higher order polynomials of lesser frame window size (as we see in Table 5.9).

### 5.6.3 Method C

In this experiment we wanted to observe the performance of the system when trained with just concatenating large number of frames without transforming the input space to a Polynomial Kernel Space. Though the concatenated frames results in a huge dimension there exist no cross correlated polynomial basis terms in the new frame. It consist of only large number of features in the original linear input space.

This means that the binary classifier being trained with such linear data will be actually trying to classify two classes of non linear data. Hence there will be high chances of misclassification resulting in poor recognition performance.

The following Table 5.10 shows the performance on three different window sizes: 301, 501 and 701. It also shows the dimension of the new frame after concatenation.

| **# Impostors** | **Window (N)** | **New Frame Size** | **Accuracy %** |
|---|---|---|---|
| 500 | 301 | 3612 | 20.6 |
| | 501 | 6012 | 22.0 |
| | 701 | 8412 | 18.6 |

Table 5.10: Performance of the system on **test150** that is trained using only Frame Concatenation without any Polynomial Expansion. Original frame size is 12.

In this case our best system performance **22.0%** is obtained with a window of 501 frames which is better than the initial system with order 2 polynomial expansion (**21.3%**) but poorer in comparison with order 3 (**26.0%**) and order 4 (**30.6%**) systems.

Though we are concatenating a very large number of frames but the result of the identification on **test150** seems to be poor. Further we observe that the performance increases as we increased the window size from 301 to 501. We then increased the window size to 701 hoping to get more improvement but the performance was degraded with this window size in comparison to previous window with 501 frames. This shows an instance of overfitting.

Hence the optimal number of window size needs to be chosen carefully which can be obtained by doing lots of experiments and trying (guessing) with different random numbers (window size) and find the point where the system gives the optimal performance. However due to constraint in time, we could not run more experiments on this method varying the window size to draw more useful observations.

## 5.7 Summary

We replicated the identification system as described in [8] and found best performance with polynomial order 4. However GMM-UBM baseline system outperformed all our system performance. Different experiments were performed to understand the significance of length of training and test data. Also experiments were carried out to see the effect of different impostor population in recognition performance. The proposed methodology involving concatenation of neighboring frames were also implemented as discussed in the chapter.

# Chapter 6

# Conclusion

The project was successfully implemented using Kernel Machines, Polynomial Kernel in particular. A small subset of the NIST SRE 2008 Corpus consisting of only telephonic speech was used as our experimental database. A detailed literature on Speaker Recognition was presented where we described various features commonly used, different Speaker Modeling Techniques and the Current State of the Art Technique, the i-Vectors. Some of the common challenges and problems often encountered in the field were pointed out. Current and future research trends along with some of the commonly used software tools in Speaker Recognition were also discussed briefly. Then a detailed description of the methodology based on Kernel Machines for Speaker Classification was presented. Various steps needed to be followed for implementing the System were discussed. Our proposed methodology that aims to enhance the performance of the Identification System by including more wider temporal context through Frame Concatenation were presented. The baseline GMM-UBM System was developed using the ALIZE toolkit. 500 impostors and 2048 gaussian mixture components were used. Different experiments were performed in Chapter 5 that examined several aspects of our Speaker Identification System based on Kernel Machines. Some of the key observation and conclusion can be stated as given below:

- An increase in polynomial order increased the performance of the Identification system. This is due to the fact that the number of polynomial basis terms (features) increases during polynomial expansion.
- Order 4 Systems seem to provide better results in comparison to Order 2 and Order 3. However we found Order 3 systems to be more optimal one as it gives recognition accuracy that is closer to Order 4 systems while incurring very less computational cost.
- This also gives an intuition that as we keep increasing the polynomial order the model starts overfitting.

- The performance of the Identification System improves when unseen data is used to train impostor model. Also experiments showed that use of large number of unseen data in training impostor model increases the performance accuracy.
- Experiments using different length training and test data showed that its very important to have sufficient data in training and testing the Speaker models to achieve a good degree of recognition performance on unseen data.
- Normalization of the classifier score by compensating with training data length showed 2% improvement in accuracy over the original system (as shown in Table 5.5).
- The methodology [8] works well on clean speech however it is very sensitive to noisy speech. Authors in [8][15] have mostly used this methodology on a clean speech YOHO database [19] that consist of a fixed length combinational lock phrase (eg. "22-34-45").
- YOHO database are specifically designed for doing experiments on Speaker Recognition. However we did our experiments on telephonic speech data which is very noisy. This is one of the key reason why our systems did not work well as the data being used in itself was a big challenge.
- Proposed Method A did not work because the polynomial expansion was done on the concatenated features that consist of only the same dimensional terms of the N window frames. Therefore, we were losing the cross correlation across original dimensional terms. As a consequence the recognition performance was very poor.
- Method C also did not work despite concatenating a huge number of frames (we tried upto 700). This is because no polynomial expansion is performed here, and the high dimensional frame consists of only the original features in linear input space.
- Method B on the contrary improved the recognition results. It is observed that with the use of more larger window the identification accuracy increases however the computational cost involved also increases dramatically.
- The main challenge and difficulty in this project was to manage the huge computational cost involved with an increase in Order and window size.
- Kernel Trick could not be implemented in our project because we were losing the actual order of similarity measure for computation of Correlation Matrix. Rather than computing the similarity measure of the features across time the Kernel Trick would compute the similarity measure of features over the entire dimension.

Also the baseline GMM-UBM system outperformed the performance of all our systems that is based on Kernel Machines because of the above mentioned reasons.

# Bibliography


[1] Cholesky decomposition - wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Cholesky_decomposition. Accessed: 03-09-2014.

[2] HTK speech recognition toolkit. http://htk.eng.cam.ac.uk/. Accessed: 22-08-2014.

[3] NIST speech group website. http://www.itl.nist.gov/iad/mig//tests/sre/2008/. Accessed: 22-08-2014.

[4] Polynomial kernel - wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Polynomial_kernel. Accessed: 06-09-2014.

[5] SPro home page. http://www.irisa.fr/metiss/guig/spro/. Accessed: 22-08-2014.

[6] SVM-light support vector machine. http://svmlight.joachims.org/. Accessed: 22-08-2014.

[7] SVMTorch. http://bengio.abracadoudou.com/SVMTorch.html. Accessed: 22-08-2014.

[8] Khaled T Assaleh and William M Campbell. Speaker identification using a polynomial-based classifier. In *Signal Processing and Its Applications, 1999. ISSPA'99. Proceedings of the Fifth International Symposium on*, volume 1, pages 115–118. IEEE, 1999.

[9] Bishnu S Atal. Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification. *the Journal of the Acoustical Society of America*, 55(6):1304–1312, 1974.

[10] Roland Auckenthaler, Michael Carey, and Harvey Lloyd-Thomas. Score normalization for text-independent speaker verification systems. *Digital Signal Processing*, 10(1):42–54, 2000.

[11] Christopher M Bishop et al. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[12] Herve A Bourlard and Nelson Morgan. *Connectionist speech recognition: a hybrid approach*, volume 247. Springer, 1994.

[13] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[14] William M Campbell. A sequence kernel and its application to speaker recognition. In *Advances in Neural Information Processing Systems*, pages 1157–1163, 2001.

[15] William M Campbell and Khaled T Assaleh. Polynomial classifier techniques for speaker verification. In *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*, volume 1, pages 321–324. IEEE, 1999.

[16] William M Campbell, Khaled T Assaleh, and Charles C Broun. Speaker recognition with polynomial classifiers. *Speech and Audio Processing, IEEE Transactions on*, 10(4):205–212, 2002.

[17] William M Campbell, Joseph P Campbell, Douglas A Reynolds, Elliot Singer, and Pedro A Torres-Carrasquillo. Support vector machines for speaker and language recognition. *Computer Speech & Language*, 20(2):210–229, 2006.

[18] William M Campbell, Douglas E Sturim, and Douglas A Reynolds. Support vector machines using gmm supervectors for speaker verification. *Signal Processing Letters, IEEE*, 13(5):308–311, 2006.

[19] Joseph P Campbell Jr. Testing with the yoho cd-rom voice verification corpus. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 341–344. IEEE, 1995.

[20] Joseph P Campbell Jr. Speaker recognition: a tutorial. *Proceedings of the IEEE*, 85(9):1437–1462, 1997.

[21] Joseph P Campbell Jr and Douglas A Reynolds. Corpora for the evaluation of speaker recognition systems. In *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*, volume 2, pages 829–832. IEEE, 1999.

[22] KH Davis, R Biddulph, and S Balashek. Automatic recognition of spoken digits. *The Journal of the Acoustical Society of America*, 24(6):637–642, 1952.

[23] Steven Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 28(4):357–366, 1980.

[24] Najim Dehak, Patrick Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet. Front-end factor analysis for speaker verification. *Audio, Speech, and Language Processing, IEEE Transactions on*, 19(4):788–798, 2011.

[25] George R Doddington, Mark A Przybocki, Alvin F Martin, and Douglas A Reynolds. The nist speaker recognition evaluation–overview, methodology, systems, results, perspective. *Speech Communication*, 31(2):225–254, 2000.

[26] RB Dunn, TF Quatieri, DA Reynolds, and JP Campbell. Speaker recognition from coded speech and the effects of score normalization. In *Signals, Systems and Computers, 2001. Conference Record of the Thirty-Fifth Asilomar Conference on*, volume 2, pages 1562–1567. IEEE, 2001.

[27] Sadaoki Furui. Fifty years of progress in speech and speaker recognition. *The Journal of the Acoustical Society of America*, 116(4):2497–2498, 2004.

[28] Gene H Golub and Charles F van Van Loan. Matrix computations (johns hopkins studies in mathematical sciences). 1996.

[29] JA Haigh and JS Mason. Robust voice activity detection using cepstral features. In *TENCON'93. Proceedings. Computer, Communication, Control and Power Engineering. 1993 IEEE Region 10 Conference on*, pages 321–324. IEEE, 1993.

[30] Hynek Hermansky. Perceptual linear predictive (plp) analysis of speech. *the Journal of the Acoustical Society of America*, 87(4):1738–1752, 1990.

[31] Hynek Hermansky and Nelson Morgan. Rasta processing of speech. *Speech and Audio Processing, IEEE Transactions on*, 2(4):578–589, 1994.

[32] Patrick Kenny, Gilles Boulianne, Pierre Ouellet, and Pierre Dumouchel. Joint factor analysis versus eigenchannels in speaker recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 15(4):1435–1447, 2007.

[33] Patrick Kenny, Pierre Ouellet, Najim Dehak, Vishwa Gupta, and Pierre Dumouchel. A study of interspeaker variability in speaker verification. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(5):980–988, 2008.

[34] Tomi Kinnunen and Haizhou Li. An overview of text-independent speaker recognition: from features to supervectors. *Speech communication*, 52(1):12–40, 2010.

[35] Anthony Larcher, Jean-Francois Bonastre, Benoit GB Fauve, Kong-Aik Lee, Christophe Lévy, Haizhou Li, John SD Mason, and Jean-Yves Parfait. Alize 3.0-open source toolkit for state-of-the-art speaker recognition. In *INTERSPEECH*, pages 2768–2772, 2013.

[36] John Makhoul. Linear prediction: A tutorial review. *Proceedings of the IEEE*, 63(4):561–580, 1975.

[37] S Matsoukas, R Iyer, O Kimball, J Ma, T Colthurst, R Prasad, and CL Kao. Bbn cts english system. In *NIST RT-03 Workshop*, 2003.

[38] Tomoko Matsui and Sadaoki Furui. Comparison of text-independent speaker recognition methods using vq-distortion and discrete/continuous hmm's. *Speech and Audio Processing, IEEE Transactions on*, 2(3):456–459, 1994.

[39] J Oglesby and JS Mason. Radial basis function networks for speaker recognition. In *Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on*, pages 393–396. IEEE, 1991.

[40] Jason Pelecanos and Sridha Sridharan. Feature warping for robust speaker verification. 2001.

[41] Douglas A Reynolds. Experimental evaluation of features for robust speaker identification. *Speech and Audio Processing, IEEE Transactions on*, 2(4):639–643, 1994.

[42] Douglas A Reynolds. Speaker identification and verification using gaussian mixture speaker models. *Speech communication*, 17(1):91–108, 1995.

[43] Douglas A Reynolds. Comparison of background normalization methods for text-independent speaker verification. In *Eurospeech*, 1997.

[44] Douglas A Reynolds. Channel robust speaker verification via feature mapping. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 2, pages II–53. IEEE, 2003.

[45] Douglas A Reynolds, Thomas F Quatieri, and Robert B Dunn. Speaker verification using adapted gaussian mixture models. *Digital signal processing*, 10(1):19–41, 2000.

[46] Douglas A Reynolds and Richard C Rose. Robust text-independent speaker identification using gaussian mixture speaker models. *Speech and Audio Processing, IEEE Transactions on*, 3(1):72–83, 1995.

[47] Laszlo Rudasi and Stephen A Zahorian. Text-independent talker identification with neural networks. In *Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on*, pages 389–392. IEEE, 1991.

[48] Roberto Togneri and Daniel Pullella. An overview of speaker identification: Accuracy and robustness issues. *Circuits and Systems Magazine, IEEE*, 11(2):23–61, 2011.

[49] V Wan and WM Campbell. Support vector machines for speaker verification and identification. In *Neural Networks for Signal Processing X, 2000. Proceedings of the 2000 IEEE Signal Processing Society Workshop*, volume 2, pages 775–784. IEEE, 2000.